CGL Technical Report, No. TP-2008-00, November 2008

Typesetting CGL technical reports
The `cglarticle` LaTeXclass

Panagiotis B. Perakis
Computer Graphics Laboratory
Department of Informatics and Telecommunications
University of Athens, GREECE

Abstract This report is a user guide for typesetting Computer Graphics Laboratory technical reports using the `cglarticle` LaTeX class definition. It also offers an introduction to TeX and LaTeX typesetting systems. Instructions for using the `cglarticle` class are included in this document.

Keywords Tex, LaTex, typesetting, article, technical report.

Version 0.1b This report is a beta version. Bug reports, comments, improvements etc. are welcomed at `takis@antinoos.gr`.

Computer Graphics Laboratory
Department of Informatics and Telecommunications
University of Athens
15784 Ilisia
GREECE
http://graphics.di.uoa.gr

Contents

1 Introduction

TEX and LATEX are scientific typesetting systems that offer programmable desktop publishing features and extensive facilities for automating most aspects of typesetting and desktop publishing of certain documents easier, especially those documents containing equations, figures, tables, indices and large bibliographies, including numbering, cross-referencing and page layout.

TEX and LATEX are most widely used by mathematicians, scientists, engineers, philosophers, scholars in academia and the commercial world, and other professionals.

TEX was designed for the UNIX operating system, but versions of it are available for Windows machines.

A nice thing about this system is its portability. A document prepared in this system can be transported to any machine having any operating system, so long as the machine has TEX and/or LATEX installed. The portability works at two levels. First, one can have the raw document that one types in (usually having a filename ending in `.tex`) transported to any machine and compiled on it. The second level is, on compilation the machine produces a file with extension `.dvi` which stands for device-independent file or `.pdf` for portable document format. These files can be transported to any machine and used to print or view the document. The second advantage with these systems is that they allow one to control the layout of the document as would be done by a type-setter.

TEX (pronounced /*tɛx*/, as in Greek, often /*tɛk*/ in English; written with a lowercase 'e' in imitation of the logo) is a typesetting system designed and mostly written by Donald Knuth [3, 4]. The name of TeX derives from the Greek word $\tau\epsilon\chi\nu\eta$ (skill, art, technique); for this reason, TEX's creator Donald Knuth promotes a /*tɛx*/ pronunciation as in Modern Greek.

TEX was designed with two main goals in mind: to allow anybody to produce high-quality books using a reasonable amount of effort, and to provide a system that would give the exact same results on all computers, now and in the future.

Within the typesetting system, its name is styled as TEX. TEX is free software.

LATEX (pronounced /*leɪtɛx*/ or /*leɪtɛk*/) is a document markup language and document preparation system for the TEX typesetting program.

LATEX was originally written in the early 1980s by Leslie Lamport at SRI International [5, 6]. It has become the dominant method for using TEX; relatively few people write in plain TEX anymore. The current version is LATEX2e.

LATEX is intended to provide a high-level language that accesses the power of TEX. LATEX essentially comprises a collection of TEX macros and a program to process LATEX documents. Because the TEX formatting commands are very low-level, it is usually much simpler for end-users to use LATEX.

Within the typesetting system, its name is styled as LATEX. LATEX is free software.

The term LATEX refers only to the language in which documents are written, not to the text editor itself. In order to create a document in LATEX, a `.tex` file must be created using some form of text editor. While many text editors work, many people prefer to use one of several editors designed specifically for working with LATEX (such as WinEdt for Windows (Chap.: 2)).

1.1 The TEX typesetting system

TEX commands commonly start with a backslash and are grouped with curly braces. However, almost all of TEX's syntactic properties can be changed on the fly which makes TEX input hard to parse by anything but TEX itself. TEX is a macro- and token-based language: many commands, including most user-defined ones, are expanded on the fly until only unexpandable tokens remain which get executed. Expansion itself is practically side-effect free. Tail recursion of macros takes no memory, and if-then-else constructs are available. This makes TEX a *Turing-complete language* even at expansion level.

The system can be divided into four levels:

- in the first, characters are read from the input file and assigned a category code (sometimes called "catcode", for short). Combinations of a backslash followed by letters or a single other character are replaced by a control sequence token. In this sense this stage is like lexical analysis.

- In the next stage, expandable control sequences (such as conditionals or defined macros) are replaced by their replacement text.

- The input for the third stage is then a stream of characters (including ones with special meaning) and unexpandable control sequences (typically assignments and visual commands). Here characters get assembled into a paragraph. TEX's paragraph breaking algorithm works by optimizing breakpoints over the whole paragraph.

- The fourth stage breaks the vertical list of lines and other material into pages.

The TEX system has precise knowledge of the sizes of all characters and symbols, and using this information, it computes the optimal arrangement of letters per line and lines per page. It then produces a DVI file ("DeVice Independent") containing the final locations of all characters. This `.dvi` file can be printed directly given an appropriate printer driver, or it can be converted to other formats. Nowadays, PDF-TEX is often used which bypasses DVI generation by producing a PDF file ("Portable Document Format").

The base TEX system understands about 300 commands, called *primitives*. However, these low-level commands are rarely used directly by users, and most functionality is provided by format files. Knuth's original default format, which adds about 600 commands, is Plain TEX. The most widely used format is LATEX originally developed by Leslie Lamport, which incorporates document styles for books, letters, slides, etc, and adds support for referencing and automatic numbering of sections and equations. Another widely used format, AMS-TEX is produced by the American Mathematical Society, and provides many more user-friendly commands, which can be altered by journals to fit with their house style. Most of the features of AMS-TEX can be used in LATEX by using the AMS "packages". This is then referred to as AMS-LATEX (Chap.: 3)).

1.2 The LaTeX typesetting system

LaTeX is based on the idea that authors should be able to focus on the content of what they are writing without being distracted by its visual presentation. In preparing a LaTeX document, the author specifies the logical structure using familiar concepts such as chapter, section, table, figure, etc., and lets the LaTeX system worry about the presentation of these structures. It therefore encourages the separation of layout from content, while still allowing manual typesetting adjustments where needed. This is similar to the mechanism by which many word processors allow styles to be defined globally for an entire document or the CSS mechanism used by HTML.

LaTeX can be arbitrarily extended by using the underlying macro language to develop custom formats. Such macros are often collected into "packages", which are available to address special formatting issues such as complicated mathematical content or graphics.

1.3 Classes and packages of LaTeX 2e

This section covers some general points concerned with writing LaTeX classes and packages [7].

LaTeX is a document preparation system that enables the document writer to concentrate on the contents of their text, without bothering too much about. the formatting of it. For example, chapters are indicated by `\chapter{<title>}` rather than by selecting `18pt bold`. The file that contains the information about how to turn logical structure (like '`\chapter`') into formatting (like '`18pt bold ragged right`') is a *document class*. In addition, some features (such as colour or included graphics) are independent of the document class and these are contained in *packages*.

One of the largest differences between LaTeX 2.09 and LaTeX 2e is in the commands used to write packages and classes. In LaTeX 2.09, there was very little support for writing *package* (`.sty`) and *class* (`.cls`) files, and so writers had to resort to using low-level commands. LaTeX 2e provides high-level commands for structuring packages. It is also much easier to build classes and packages on top of each other.

The first thing to do when you want to put some new LaTeX commands in a file is to decide whether it should be a "document class" or a "package".
The rule of thumb is:

> *If the commands could be used with any document class,*
> *then make them a package;*
> *and if not, then make them a class.*

There are two major types of class: those like article, report or letter, which are *free-standing*; and those which are *extensions* or *variations* of other classes. For example, the `cglarticle` document class, which is built on the article document class.

Thus, a company might have a local ownlet class for printing letters with their own headed note-paper. Such a class would build on top of the existing letter class, but it cannot be used with any other document class, so we have `ownlet.cls`

rather than `ownlet.sty`. The graphics package, in contrast, provides commands for including images into a LaTeX document. Since these commands can be used with any document class, we have `graphics.sty` rather than `graphics.cls`.

2 Running TeX on Windows

To download and successfully run TeX on Windows machines, you will need to get 3 different programs: (1) WinEdt, (2) Ghostscript/GSview and (3) MiKTeX.

WinEdt: WinEdt (shareware) is a powerful and versatile ASCII editor and shell for MS Windows with a strong predisposition towards the creation of LaTeX documents. WinEdt itself is not a TeX system! You'll have to download and install a (free) Win32 TeX system of your choice (MiKTeX, or TeX Live). WinEdt setup and default settings have been carefully prepared to make installation and integration with MiKTeX automatic. It is recommended to install WinEdt, MiKTeX, Ghostscript, GSView, and Adobe Reader as a power user in order to allow the programs and their installers to properly update Windows Registry and make automatic integration with WinEdt possible.
To install WinEdt, download and execute the setup file `winedt55.exe` from `http://www.winedt.com/`.

Ghostscript/GSView: Ghostscript is the name of a set of free software that provides: An interpreter for the PostScript language, with the ability to convert PostScript language files to many raster formats, view them on displays, and print them on printers that don't have PostScript language capability built in; an interpreter for Portable Document Format (PDF) files, with the same abilities; the ability to convert PostScript language files to PDF (with some limitations) and vice versa; and a set of C procedures (the Ghostscript library) that implement the graphics capabilities that appear as primitive operations in the PostScript language.

GSview is a previewer for Windows, OS/2 & Linux.

To install Ghostscript/GSView, download and execute the setup files `gs862w32.exe` (GPL Ghostscript 8.62) and `gsv49w32.exe` (GSview 4.9) for 32-bit Windows from `http://pages.cs.wisc.edu/~ghost/`.

MiKTeX: MiKTeX is an open source up-to-date implementation of TeX and related programs for Windows. The DVI previewer '`Yap`' (part of the MikTeX distribution) allows for an optimized edit-compile-view cycle. It can also display embedded `.eps` files (provided a Ghostscript interpreter is installed).
To install MiKTeX, download and execute the setup file `basic - miktex - 2.7.3107.exe` (MiKTeX 2.7 for 32-bit Windows) from `http://miktex.org/`.

3 Creating a LaTeX document

Now I will very briefly describe how one goes about making a document in LaTeX [5, 9, 7].

Prepare a file following the rules of LaTeX. The name of the file should be *foo*.tex[1]. That is, the extension should be `.tex`.

In fact, the way to use this document is to look at the `cglarticle.tex` file itself and the output of this file together. By comparison, you will be able to learn how to use different environments. The next thing is to use the text of this file in your LaTeX document editor and edit appropriate portions to suit your needs.

In a TeX file the opening line declares the *style* of document one is making: `\documentclass[12pt,a4paper]{article}`.
Other document styles are `letter`, `report`, `book` and many others. One can define one's own style. The style defined here for CGL is `cglarticle.cls`: `\documentclass{cglarticle}`.
The style declaration goes in curly brackets. The parameters in square brackets are details of formatting options.

One needs to use different *packages* which do specialized jobs: `\usepackage{epsfig}`.
The package `epsfig` is required to import figures and diagrams which are in encapsulated postscript format (`.eps`) for dvi output into the document at appropriate place. You may use `\usepackage{graphicx}`, as a standard LaTeX graphics tool when including `.jpg` figure files for pdf output (Chap.: 3.6).

If you want to use all the symbols in your LaTeX Document you should load the `amsmath` package in your preamble:

```
\usepackage{amsmath, amsxtra, amstext, amssymb}
\usepackage{latexsym}
\usepackage{dsfont} % for \mathds{N}
```

The first line of the document file that is executed is the `\begin{document}` line. This tells that what follows is the document to be processed by TeX. The last line of the document is `\end{document}`. Whatever that follows this line is not processed by TeX.

You can also define *macros* (shortcuts) which are used in preparing the document:

```
\newcommand{\tx} {\TeX$\;$}
\newcommand{\ltx} {\LaTeX$\;$}
```

Here `\tx` is equivalent to TeX. The macros are useful when one needs to type certain things again and again. So one can use a short-cut to represent the full thing in the text.

Before we go on, let me mention that there are some *special characters* in TeX and LaTeX . Some of these are `%`, `&`, `$`, `{`, `}`, `_` and `^`. Their meanings are defined in the table below.

[1] Here *foo* is a generic name of a file. So, replace *foo* by the name of the file you are dealing with.

`%`	Comment. Whatever follows % on the line is ignored
`&`	Character used for separating columns in tables
`$`	Character to signal begining and ending of math mode. Whatever is present between two $'s or two pairs of $'s (i.e. $$ ⋯ $$) is in math mode
`{` and `}`	Characters used to define the extent of an environment. { and } are not printed.
`_` and `^`	characters used to define subscripts and superscripts in math environment

3.1 Text Environment

The most elementary environment is the *text environment*. Normally, the text you want to print should be placed on a page and can be left or right justified. That is, the left-most and right-most parts of the text should be aligned. One need not declare the text environment since it is the standard environment.

You may want to *change fonts* within the text. The number of fonts available is somewhat limited but these are enough for a scientific document. The standard text is displayed in roman font. Other often used fonts and typefaces are shown in the table here. The left column shows the type of font used, the middle column shows how the text is entered in the latex file and the right column shows how it appears on the page. Note that the text written in the changed fonts is enclosed in the braces.

emphasize	\emph{emphasize}	*emphasize*
bold face	\bf{bold face}	**bold face**
typewriter	\tt{typewriter}	`typewriter`
small capitals	\sc{small capitals}	SMALL CAPITALS
italic	\it{italic }	*italic*
slanted	\sl{slanted}	*slanted*
roman	\rm{roman}	roman

In addition to the different fonts, the size of the fonts can also be changed in the document. For example, the text of a document is prepared in the fonts of 12 points (declared in the `\documentclass` statement)[2]. One can use different font sizes. One way is to change the font size in the `\documentclass` statement itself. That will change the size of all characters in the document. The other is to scale the size of the letters with respect to the 'default' font size one is using. Different sizes available are tiny, scriptsize, footnotesize, small, normalsize, large, Large, LARGE, huge and Huge. So there is a large variation of font sizes one can have.

You may want to have your document justified at left or at right or centered. This is done by `flushleft`, `flushright` and `center` environments.

[2]This is the unit in which the typesetter measures length.

Below is an example of doing it:

Left justified text
This text is left justified. Therefore the right margin is uneven, as would appear when you type the manuscript normally. This is the *flushleft* environment. This effect is produced by inserting this text in between the statements `\begin{flushleft}` and `\end{flushleft}`.

Right justified text
This text is right justified. Therefore the left margin is uneven. This is the *flushright* environment, used in paragraph mode. It is invoked by inserting this text in between the statements `\begin{flushright}` and `\end{flushright}`.

Centered Text
Here the paragraph is centered. You need this environment when typing titles. Again, this is in paragraph mode and the effect is produced by inserting this text in between the statements `\begin{center}` and `\end{center}`.

Another way to change the document's justification is to use `\raggedright`, `\raggedleft` and `\centering` declarations respectively.

If you want to change line you can use the `\\` or the `\linebreak` command. The first forces a new line but the text of the line is left justified and has minimum width, since the second forces a new line but the text of the line is spread over the whole text width.

3.2 Math Environment

Now we come to the most useful environment for which LaTeX or TeX has edge over all word processors. That is the *mathematical environment*. There are various types of mathematical texts. One situation is when you want to insert a formula within the text (appearing on the same line). Another situation is when you want the mathematical expression to appear on a separate line but not as a regular equation (with equation number etc.). Finally you may want a regular equation. Usually, you would want to format the equations so that they look aesthetically pleasing. You can do all that in math environment. Simple in-line expressions are written between two `$` signs. For example, you can write `$x=y$` to get $x = y$. Note the change in the font of the mathematical equation. If you want to write this equation on a separate line, you can write `$$x=y$$` to get

$$x = y$$

Coming to regular equations, there are two environments possible. These are `equation` and `eqnarray`. The *equation environment* is useful for single, short equations which do not require more than one line. For example, you can write:

```
\begin{equation}
  \label{eq:1}
  y=e^x
\end{equation}
```

to get:

$$y = e^x \tag{1}$$

Note that the *equation environment* is quite similar to the earlier example except that one now has equation number appearing on the right. One can also label the equations for reference in the text. For example, if one wants to say that y is a monotonically increasing function of x because of eq(1), one would write `...because of eq(\ref{eq:1})`.

Finally, *eqnarray environment* is a combination of equation and array environments. It is useful when the equations are long and overflow to many lines or when one wants to write several equations together so that they look aesthetically pleasing. For example, one may want to align = signs in one column. Below is an example of doing this. You can write:

```
\begin{eqnarray}
  \label{eq:2}
  y & = & \frac{e^{ix} - e^{-ix}}{2i} \\
    & = & \sin(x)
\end{eqnarray}
```

to get:

$$\begin{aligned} y &= \frac{e^{ix} - e^{-ix}}{2i} && (2)\\ &= \sin(x) && (3) \end{aligned}$$

In *eqnarray environment*, each line is treated as an equation and numbered. If you are writing a long equation, each line should not be numbered. This is done by typing `\nonumber` before `\\`. Note that `\\` ends a line in text as well as math environment.

You have already seen how to write a fraction using the macro `\frac`. A number of such macros are defined in LaTeX and these are quite handy. These are too many to enumerate here.

You can also write equations on a separate line by using the `\[...\]` delimeters. So, you can write:

```
\[D^{2}=\sum_{i=1}^{N}{\alpha_{i}|\vec{x_{i}}-\vec{x_{m}}|^{2}}\]
```

to get:

$$D^2 = \sum_{i=1}^{N} \alpha_i |\mathbf{x_i} - \mathbf{x_m}|^2$$

Other handy things are *Greek letters* which are so often used in equations. One also uses script letters (also called *calligraphic* letters) in mathematical equations. These are invoked by calligraphic declaration `\cal`. Thus we have $\mathcal{CALLIGRAPHIC}$. Note that calligraphic declaration works in math environment only.

3.3 Array Environment

One needs to write arrays in mathematical equations. One needs *array environment* for this. Array environment is also required for presenting data in a tabular form. But array environment is more versatile than the tabular environment in certain respects. We shall consider tabular environment later. The array environment is operative within math environment where as the tabular environment is operative in text environment. Let us consider that we want to write an equation $AX = Y$ explicitly. Here A is a 3×3 matrix and X and Y are 3×1 vectors. This equation is written as:

```
$$
\left (
  \begin{array}{ccc}
     a_{11} & a_{12} & a_{13} \\
     a_{21} & a_{22} & a_{23} \\
     a_{31} & a_{32} & a_{33}
  \end{array}
\right )
\left [
  \begin{array}{c}
     x_1 \\ x_2 \\ x_3
  \end{array}
\right ]
=
\left |
  \begin{array}{c}
     y_1 \\ y_2 \\ y_3
  \end{array}
\right |
$$
```

to get:

$$\left (\begin{array}{ccc} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{array} \right) \left [\begin{array}{c} x_1 \\ x_2 \\ x_3 \end{array} \right] = \left | \begin{array}{c} y_1 \\ y_2 \\ y_3 \end{array} \right |$$

I have used three different array delimiters (for matrices, vectors and determinants), as an example. Note that the delimiters are defined by means of the declaration `\left (,[,|` and `\right ),],|`. If one does not want to have one of the delimiters, one must write `\left .` and `\right .`.

3.4 Tabular Environment and Tables

There are two environments, *tabbing* and *tabular*, for producing a tabular output. I shall discuss tabular environment alone since I find it more useful and I can get away with it practically all the time. As mentioned earlier, tabular environment is similar to the array environment. The tabular environment is more suitable for text

material. Let us make a table of physical constants to illustrate the making of a table. The latex file is as shown below:

```
\begin{table}[h]
\caption{A list of some of the physical constants}
\label{tbl:1}
\begin{center}
\begin{tabular}{ll|cr}
\hline
Physical constant & name & value & dimension \\
\hline
c & velocity of light & $2.998 \times 10^8$ & m/sec \\
e & electronic charge & $1.602 \times 10^{-19}$ & coul \\
$N_A$ & Avogadro number & $6.02 \times 10^{22}$ & per mole \\
$m_e$ & Electron mass & $9.11 \times 10^{-31}$ &  kg \\
\hline
\end{tabular}
\end{center}
\end{table}
```

and the table appears as shown in Table(1).

Table 1: A list of some of the physical constants

Physical constant	name	value	dimension
c	velocity of light	2.998×10^8	m/sec
e	electronic charge	1.602×10^{-19}	coul
N_A	Avogadro number	6.02×10^{22}	per mole
m_e	Electron mass	9.11×10^{-31}	kg

One can note the similarity between the tabular and array environments. In *table environment* one can give the `\caption` or title of the table and can refer to it else where in the text, by using the `\label` macro.

Also note that one has tabular environment inside the table environment. When one declares the tabular environment, one declares the number of columns and how the contents of the columns are placed. This is given in the `\begin{tabular}{ll|cr}` macro. Thus, in the table above, we have four items in each row and the first two are left justified (`l`), the third is centered (`c`) and the last is right justified (`r`). Vertical lines are drawn by inserting `|` at appropriate places. Horizontal lines are drawn by `\hline` macro.

Notice that center environment (`\begin{center}` ... `\end{center}`) is invoked within table environment. This ensures that the table is centered within page. If we omit the center environment we will get a left justified table.

The table made below is more complex because in this table we have different number of columns for different rows. Essentially, what we have is, some columns

in a row are spanning more than one columns. For example, the first row has two columns, the first spanning one column and the second spanning three columns. This is achieved by using `\multicolumn {.}{.}` command. The latex file looks like this:

```
\begin{table}[h]
\caption{A Complicated Table}
\label{tbl:2}
\begin{center}
\begin{tabular}{|l|l|r|r|}
\hline
one column & \multicolumn{3}{c|} {three columns} \\
\hline
item1 & prop11 & prop12 & prop13 \\
\cline{2-4}
item2 & prop21 & prop22 & prop23 \\
\hline
\multicolumn{2}{|c|} {two columns} & \multicolumn{2}{l|} {two columns} \\
\hline
item3 & prop31 & prop32 & prop33 \\
\hline
\multicolumn{4} {|c|} { And So On } \\
\hline
\end{tabular}
\end{center}
\end{table}
```

and you get the Table(2).

Table 2: A Complicated Table

one column	three columns		
item1	prop11	prop12	prop13
item2	prop21	prop22	prop23
two columns		two columns	
item3	prop31	prop32	prop33
And So On			

The `[h]` in square brackets in the declaration line of the table environment means that the table is to be placed here (i.e. the place where the table environment begins). Other options are `t`, for the top of the page and `b`, for the bottom of the page.

3.5 Enumerate and Itemize Environments

Many times one has to display a list of items together. One may want to enumerate these or simply put one item on each line with some mark in front of it. The

first is generated by *enumerate* environment and the second is generated by *itemize* environment. We have encountered an example of enumerate environment earlier.

The enumerate environment is invoked by typing `\begin{enumerate}` in the first line, followed by the items you want to enumerate and closing the environment by typing `\end{enumerate}`. Each item that is to be enumerated is proceeded by the `\item` macro.

Below is a pseudo-code listing that serves as an example of itemize environment. The method of invoking the itemize environment is similar to the enumerate environment except that the word `enumerate` is replaced by `itemize`.

- Translate each example shape so that its CM is at the origin (0,0,0).
- Choose a reference shape (i.e the first example shape).
- Call this shape the reference mean shape $\mathbf{x_0}$.
- REPEAT
  - Align all example shapes to reference mean shape $\mathbf{x_0}$ by an optimal rotation
  - Recalculate the mean shape $\mathbf{x_m}$.
  - Translate the mean shape so that its CM is at the origin (0,0,0)
  - Align the mean shape $\mathbf{x_m}$ to the reference mean shape $\mathbf{x_0}$ by an optimal rotation
- UNTIL Convergence (mean shape doesn't change much): $|\mathbf{x_0} - \mathbf{x_m}| < \varepsilon$

One can also have nested enumerate and itemize environments. Usually, one does not need to have more than two nested enumerate or itemize environments but it is possible to go beyond that.

3.6 Figure Environment

Figure environment is used for including figures within the text. Encapsulated postscript files (`.eps`) can be imported into the text document. This method of figure encapsulation is **compatible with a DVI output**. (`.jpg`) files can also be imported into the text document. This method of figure encapsulation is **compatible with a PDF output**.

Below I will give two examples of including figures in a tabular environment. In order to include the eps figures in a document you must include the line `\usepackage{epsfig}` after the declaration of the `\documentclass` statement.

```
\begin{figure}[htpb]
\begin{center}
\begin{tabular}{c c}
{\epsfig{figure=images/fig1a.eps,width=0.25\textwidth}} &
{\epsfig{figure=images/fig1v.eps,width=0.25\textwidth}} \\
(a) & (b) \\
```

```
\end{tabular}
\end{center}
\caption{Caption of figures: (a) figure 1, (b) figure 2.}
\label{fig:1}
\end{figure}
```

Below is an example for including jpg files in a document. You have to include the line `\usepackage{graphicx}` after the declaration of the `\documentclass` statement.

```
\begin{figure}[htpb]
\centering
\begin{tabular}
{p{0.2\columnwidth}p{0.2\columnwidth}}
\includegraphics[width = 0.25 \textwidth]{images/fig2a.jpg} &
\includegraphics[width = 0.25 \textwidth]{images/fig2b.jpg} \\
\centering (a) & \centering (b) \\
\end{tabular}
\caption{Caption of figures: (a) figure 1, (b) figure 2.}
\label{fig:2}
\end{figure}
```

Some comments are in order. `[htpb]` in square brackets in the declaration line of the figure environment means the figure is to be placed here `h` or `t`, for the top of the page, `b` for the bottom of the page and `p` for elsewhere in page. Sizes are determined from pagewidth `\textwidth`. The name of the file containing the figure is declared in braces. This file could be in the same directory where one is working or elsewhere (i.e. `../images/`). Finally, the last two lines are figure caption and a reference label.

3.7 Cross References

In a document one often refers to other published or unpublished documents. There are different methods of giving these references and these are defined by the `\bibliographystyle{}` command. For example, in research reports and articles, the works referred to in the document are listed at the end of the document with each work being numbered or having some identifying characters (say first three letters of the author's name followed by last two digits of the year the work was published). Sometimes the references are given at the end of a chapter or at the bottom of a page (although this style of referencing is vanishing). Another use of referencing is when one is referring to an equation, table, section etc. of the same document (cross referencing). LaTeX has a nice way of handling these two cases. The method used by LaTeX takes care of the possibility of future modification of the document.

The method of *referring to other works* is as follows. The place where a work is referred, one types `\cite{xxx}`. Here `xxx` is a key to the work one is referring to. There will be many such references to different works within the body

of the text. All these references are collected and put in an external bibliography database file (`.bib`). This fiile is included into the document by using the command `\bibliography{<bib file>}`. Bib items are written in the following way, according to the bib item type (book, article, report etc.) :

```
@book{Lamport:BOOK:1994,
    author={Leslie Lamport},
    title={{L}a{T}e{X}: A Document Preparation System},
    publisher={Addison-Wesley, Reading, Massachusetts},
    edition={2nd},
    year={1994}
}
```

where `Lamport:BOOK:1994` is the item key.

Note that the key `xxx` here should be same as the key used while referring to the document in the text. This way of handling the references has two advantages. One is that the reference is referred to by means of a key which is easier to remember and type. This helps if same work is referred at many places. Second and more important advantage is that one can add the text and/or references later in the document but that does not require any change in the earlier referencing. But because of this, the text file has to be compiled twice to get the cross references correct.

For *cross referencing* equations, tables, figures etc. the method used is as follows. Any equation, table, figure etc. which is to be referred elsewhere in the text is labeled by means of a label command `\label{xxx}`. As in the proceeding paragraph, `xxx` is the key with which the object in question will be referred. The place where the object is referred, one types `\ref{xxx}`.

4 The `cglarticle` class file

The `cglarticle` style for typesetting articles is a standard LaTeX class, so any decent installation of LaTeX should be able to use it (Chap.: 2). This class file documents the `cglarticle.cls` file for typesetting CGL technical reports. The class file is based on the standard LaTeX article class. Most of the commands are therefore familiar. The `cglarticle` class should be used by CGL group members to typeset technical reports, in order to have a common layout.

A technical report is an *official publication*. Therefore it can be referred to in scientific literature. Furthermore a technical report is available to anyone asking for a copy. If your goal is *submission to a journal* and actual publication of an article, then you should first write the technical report, then the journal article and in the journal article refer to the technical report. Most often the technical report will provide more detail than available in the journal article.

A typical scientist spends a lot of time on reading the work of others and trying to unify all that in a logical framework. In many cases the efforts to learn everything there is to know in a specialized field is of interest to others as well. This work might not be publishable in journals but as a CGL technical report it may be.

When you have written software that might be of interest to others as well, a CGL technical report is the easiest way to make your efforts known and available to others.

Some of the *masters thesis* that are written by students in our group are well worth publication as a CGL technical report.

4.1 The front cover page

The information about the article to appear on the front cover should be specified before the `\maketitle` command. A typical front cover specification might look like the text depicted in Figure 1. All the commands used in this example LaTeX code will be explained in this section.

```
\title{Typesetting CGL technical reports}
\subtitle{The \texttt{cglarticle} \LaTeX class}
\author{Panagiotis B. Perakis}
\institute{\defaultaffiliation}
%\author{No second author}
%\institute{second author institute}
\runningauthor{P. Perakis}
\CGLreport{TP-2008-00}
\CGLreportdate{November 2008}
\abstract{
This report ...}
\keywords{Tex, LaTex, ...}
\version{Version 0.1b}{
    This report is a beta version ...
}
\acknowledge{
This CGL internal report style ...
}
```

Figure 1: Example titlepage decleration.

4.1.1 The title

Just as in the standard LaTeX article style, the title of the article is given with the `\title` command. Unlike the standard style, you are allowed to specify a `\subtitle` as well. The subtitle will be set in a smaller size and it will not be used in the running title.

4.1.2 The author(s)

This is where you probably need some adjustment to your standard LaTeX file. In this style file a bit of clever programming is borrowed from a Kluwer style file to enumerate all the authors of the article each with their own `\author` command.

The authors affiliation is specified with the `\institute` command. In case more then one author have the same affiliation use two (or more) consecutive `\author` specifications, followed by just one `\institute` command.

There is a default affiliation available: `\cglaff`, that looks like:

Computer Graphics Laboratory
Department of Informatics and Telecommunications
University of Athens, GREECE

So, CGL members can use this one.

4.1.3 The running author

For every CGL technical report there should be a running author. This author is specified with the `\runningingauthor` command. The name of running author is displayed in even page headers.

4.1.4 Report number and date

You can add a technical report number by the following two commands in the preamble:

```
\CGLreport{XXX}
\CGLreportdate{November 2000}
```

where `XXX` is the technical report number.

4.1.5 Report version

At the bottom of the title page the version of the article is printed. The `version` command takes two parameters. The first one is intended to describe the type or number of the version, i.e. “Submitted to:” or “Published in:” or “Draft:” or “Version 0.1b:”. The second parameter to the version command sets the details or comments on the version. The version need not be specified.

4.1.6 The abstract

The abstract is a command rather than an environment. So the entire abstract is given as the argument to the `\abstract` command.

4.1.7 The keywords

The keywords is also a command than an environment. So a list of keywords is given as the argument to the `\keywords` command.

4.2 The back cover page

The `cglarticle` class file automatically adds a back cover page to your CGL article. The inside of the cover page is the place to thank sponsors of the project you are working on.

4.2.1 Acknowledging your sponsors and/or partners

You may thank your sponsors on the inside page of the backcover. The `cglarticle` class provides the command `\acknowledge` to do so. This command works just like the `\abstract` command.

5 Epilogue

This document is the first in the CGL technical report series. At the same time it is a special issue. It is a "Quick Start" user guide for setting up TEX and LATEX typesetting systems, has the minimum necessary information for creating documents in such environments, and is also a guidance for typesetting Computer Graphics Laboratory technical reports using the `cglarticle` LATEX class definition. I hope that you find all this gathered information useful.

References

[1] P. Abrahams, K. Hargreaves, and K. Berry, *TeX for the impatient*, Free Software Foundation, 2003.

[2] R. Boomgaard, *The isisarticle LaTeX class*, Tech. report, University of Amsterdam, Feb 2001.

[3] Donald E. Knuth, *The TeXbook (Computers and Typesetting, Volume A)*, Addison-Wesley, Reading, Massachusetts, 1984.

[4] ———, *TeX: The Program (Computers and Typesetting, Volume B)*, Addison-Wesley, Reading, Massachusetts, 1986.

[5] Leslie Lamport, *LaTeX: A document preparation system*, 2nd ed., Addison-Wesley, Reading, Massachusetts, 1994.

[6] ———, *The writings of Leslie Lamport: LaTeX: A document preparation system*, Leslie Lamport's Home Page, April 2007.

[7] The LaTeX3 Project Team, *LaTeX 2e for class and package writers*, Tech. report, March 1999.

[8] ———, *LaTeX 2e for authors*, Tech. report, July 2001.

[9] S. C. Phatak, *A working guide to LaTeX*, Tech. report, Institute of Physics, Bhubaneswar, INDIA.

Acknowledgements

This CGL internal report style started its life in 2008 as a common style file to typeset CGL's articles. It was my introduction to (La)TeX programming [1, 7, 8]. The code is based on `ISIarticle` LaTeX class of Rein van den Boomgaard [2]. It is changed quite a bit to meet CGL's requirements. The section of using and programming TeX and LaTeX environments is based on a previous work of S. C. Phatak [9]. Other useful information resources were the "WIKIPEDIA" and the "LATEX3 Project Team" Documentation.
I'd like to thank all those who have available information for the subject in the Internet.